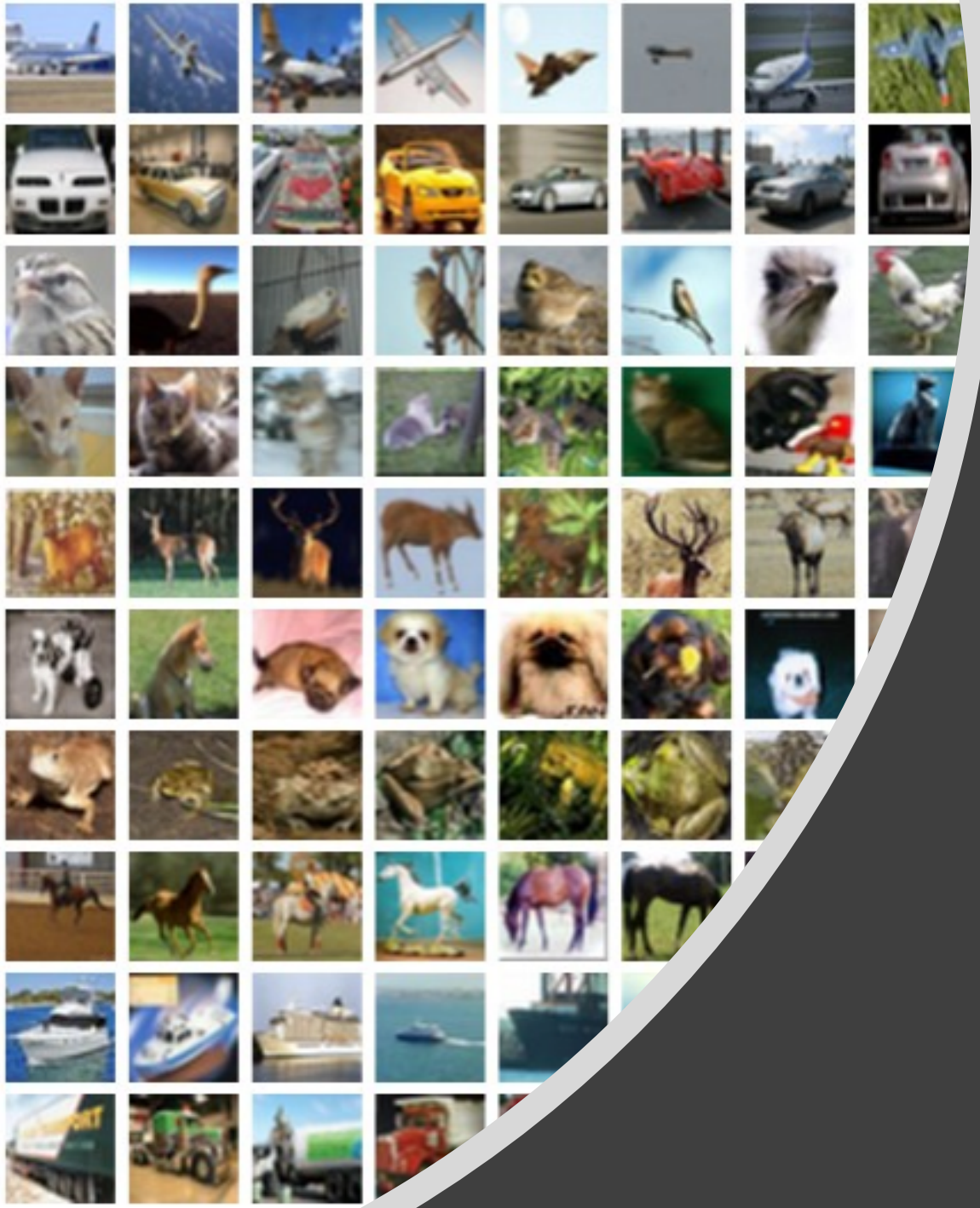
le



Dynamic Learning

Perceptron

Syllabus Update

Week 10	logistic regression, neural networks and gradient descent	Initial draft of project proposal due
Week 11	Overfitting and avoidance	Exercise 3 due
Week 12	Model Evaluation	Final draft of project proposal due
Week 13	Bayesian Methods	Exercise 4 due
Week 14	Course Recap	
Week 15	FINAL PROJECTS AND PRESENTATIONS DUE	

Final Project

- Open ended any topic involving learning models we will encounter by end of course
  - Decision trees
  - Logistic regression
  - Neural networks

- Responsibility:
  - Identify a problem you would like to work on
  - Identify source of data
  - Write the scripts to ingest the data
  - identify an appropriate Machine learning library
  - Write the scripts for passing the data to the libraries
  - Interpret the results

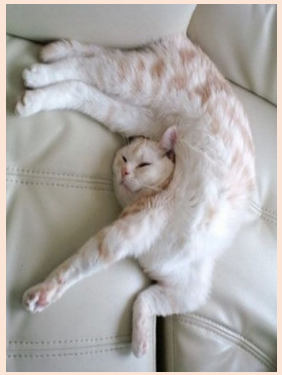
Lecture Plan

- Linear Classifiers
- Perceptron
  - Perceptron learning
- Gradient Descent
  - Stochastic Gradient Descent

Image Classification

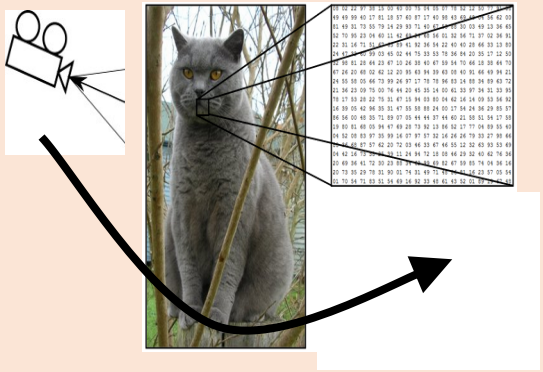
Illumination

Deformation



Occlusion

Camera pose



Background clutter

Intraclass variation

Image Classification

image parameters

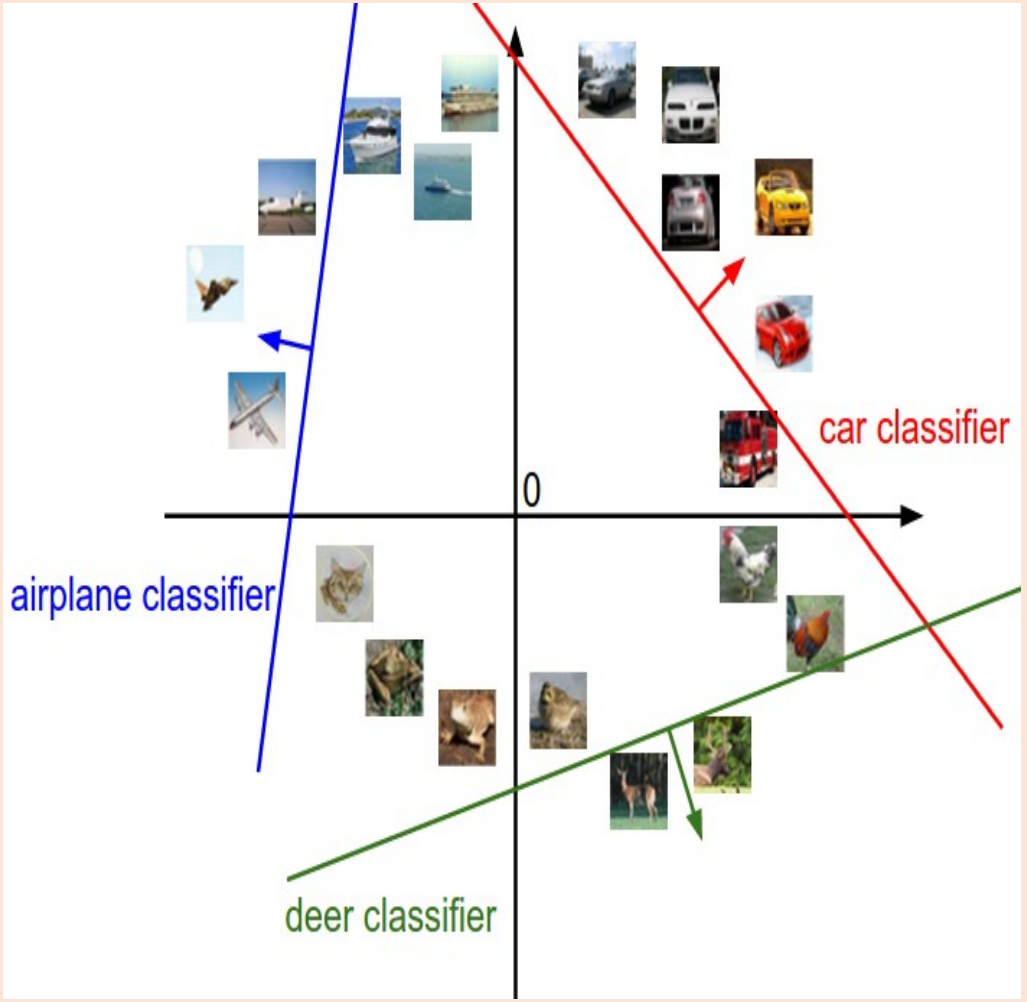
f(**x**,**W**)

classification

$$f(x_i, W, b) = Wx_i + b$$

Image Classification

$$f(x_i, W, b) = Wx_i + b$$



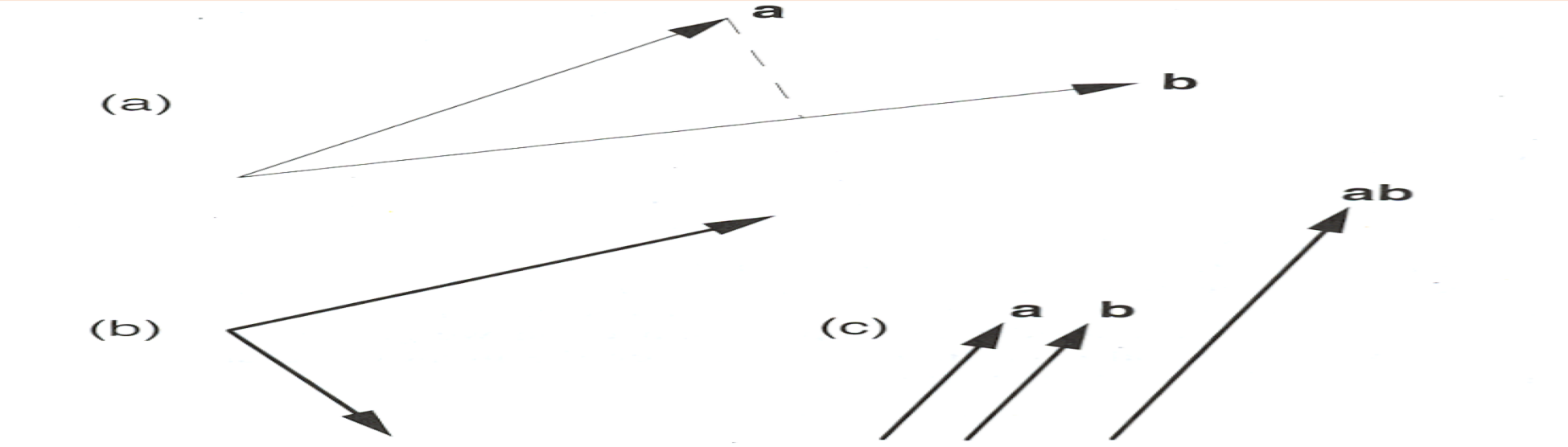
[32x32x3]
array of numbers 0...1
(3072 numbers total)

Inner Product

$$< \boldsymbol{w}, \boldsymbol{x} > = \sum_{\boldsymbol{i}} \boldsymbol{w_i} \cdot \boldsymbol{x_i}$$

$$\boldsymbol{n} =< \boldsymbol{w}, \boldsymbol{x} > = \left|\left|\boldsymbol{w}\right|\right| \cdot \left|\left|\boldsymbol{x}\right|\right| \cdot \boldsymbol{cos}(\boldsymbol{\theta})$$

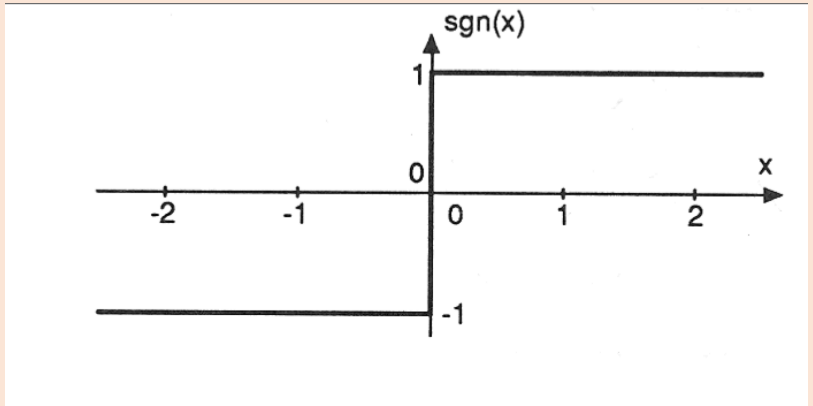
- A measure of the projection of one vector onto another



Activation Function

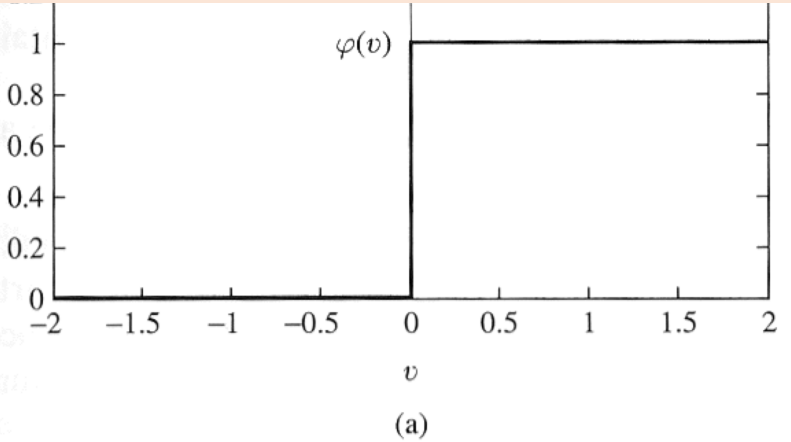
$$f(x) = sgn\left(\sum_i w_i \cdot x_i\right)$$

$$sgn(x) = \begin{cases} 1 \; if \; x \geq 0 \\ -1 \; if \; x < 0 \end{cases}$$



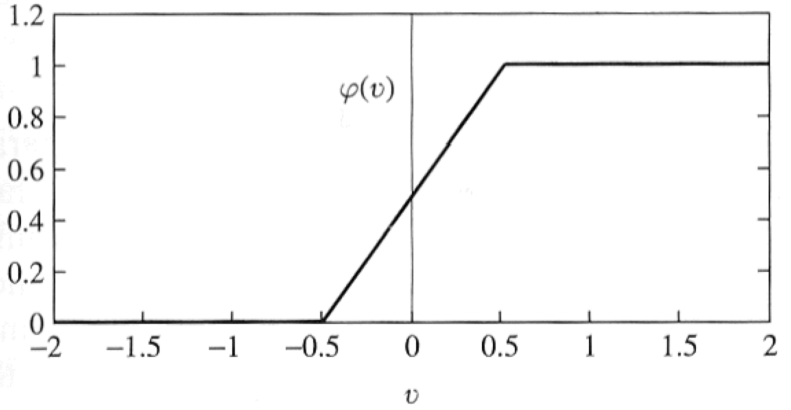
Activation Function

$$\varphi(x) = \begin{cases} 1 & if\ x \geq 0 \\ 0 & if\ x < 0 \end{cases}$$



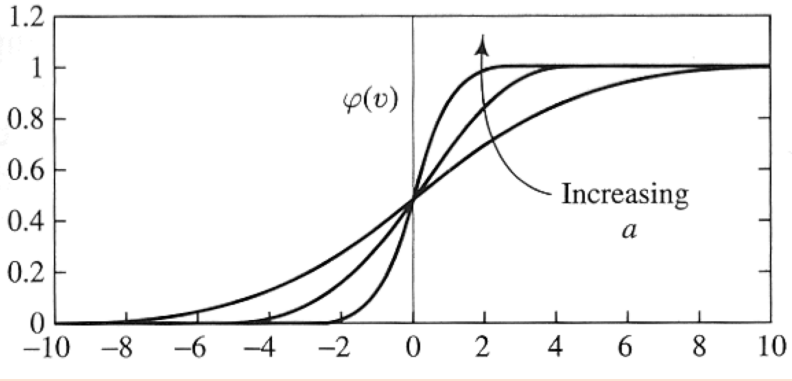
(a)

$$\varphi(x) = \begin{cases} 1 & if \quad x > 1/2 \\ x & if\ -\frac{1}{2} < x \leq 1/2 \\ -1 & if \quad x < -1/2 \end{cases}$$

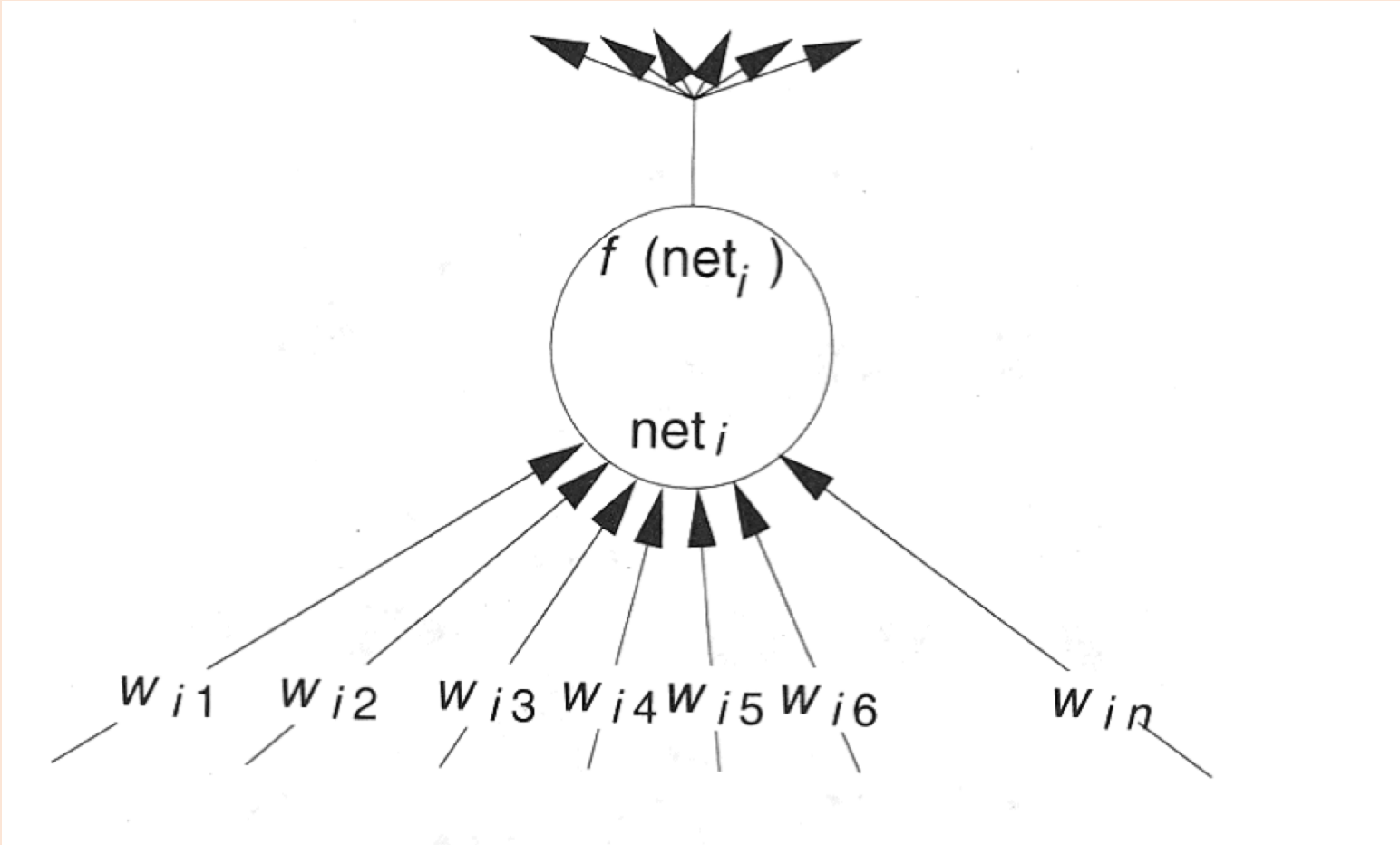


(b)

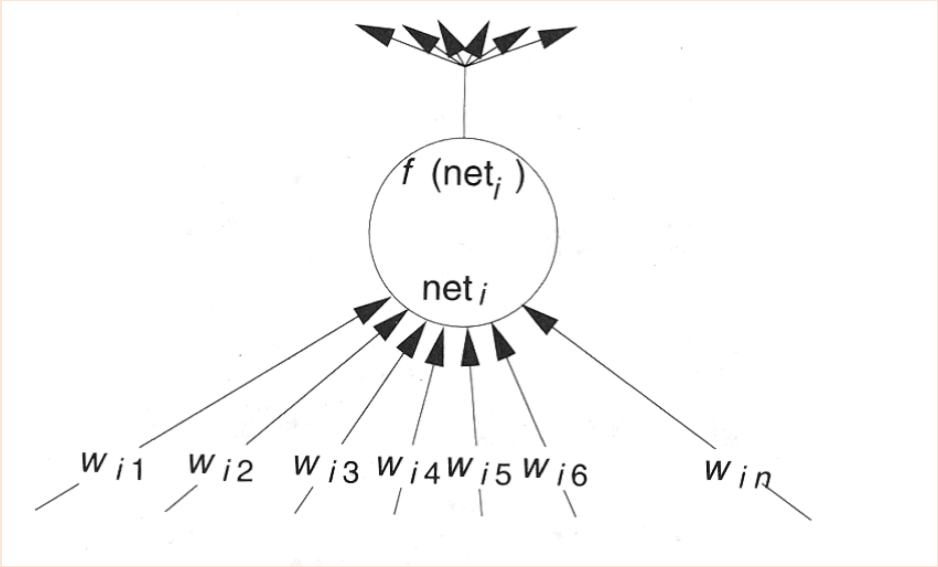
$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

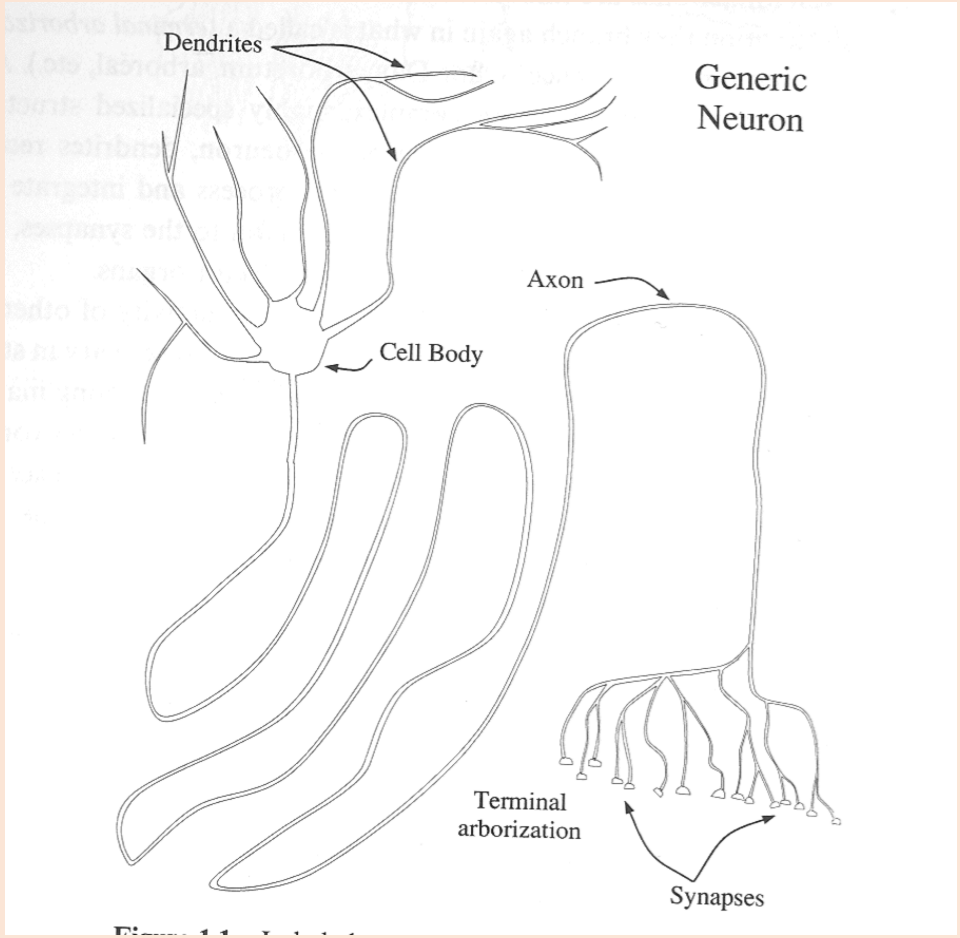


Graphical Representation



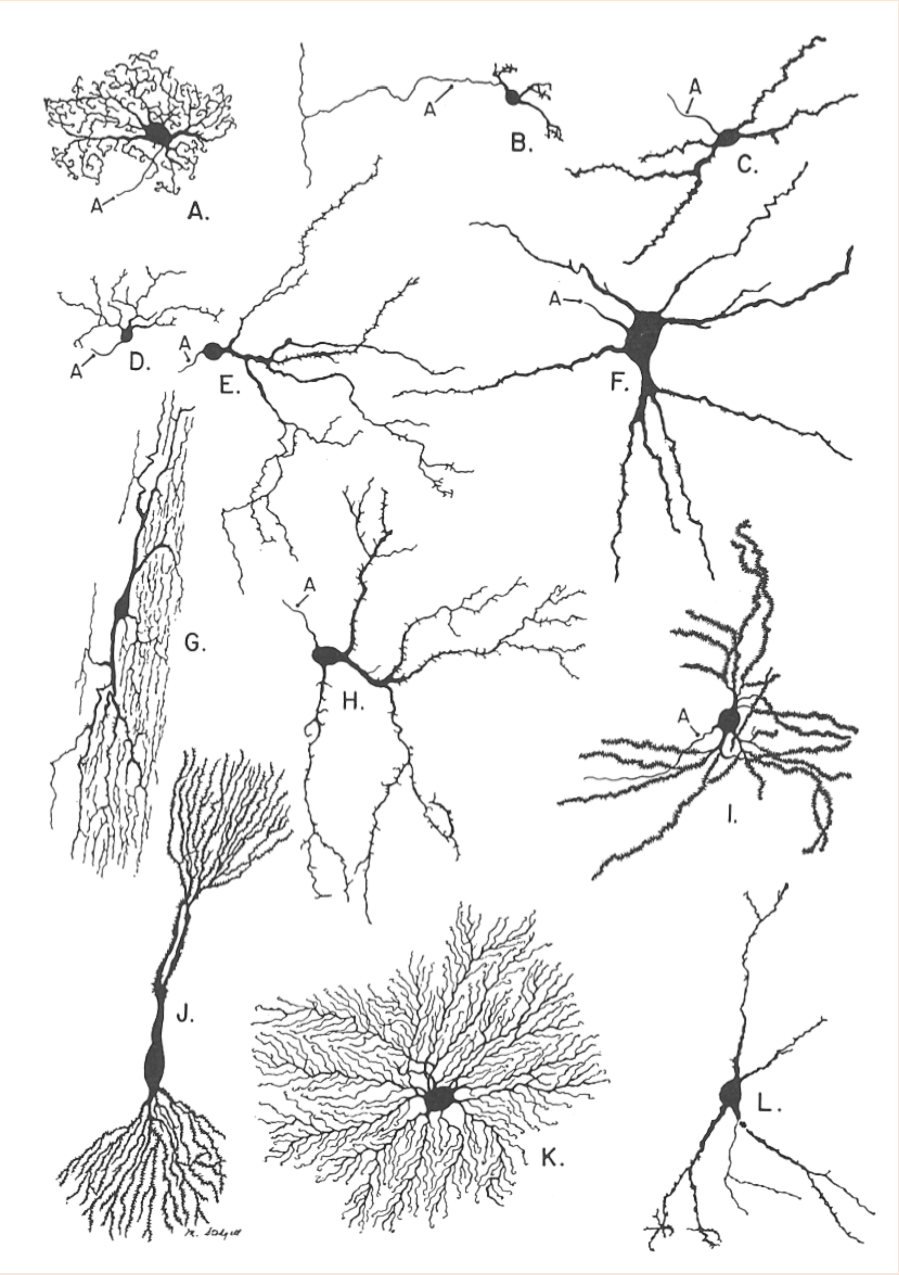
Resemblance to real neurons





Resemblance to real neurons

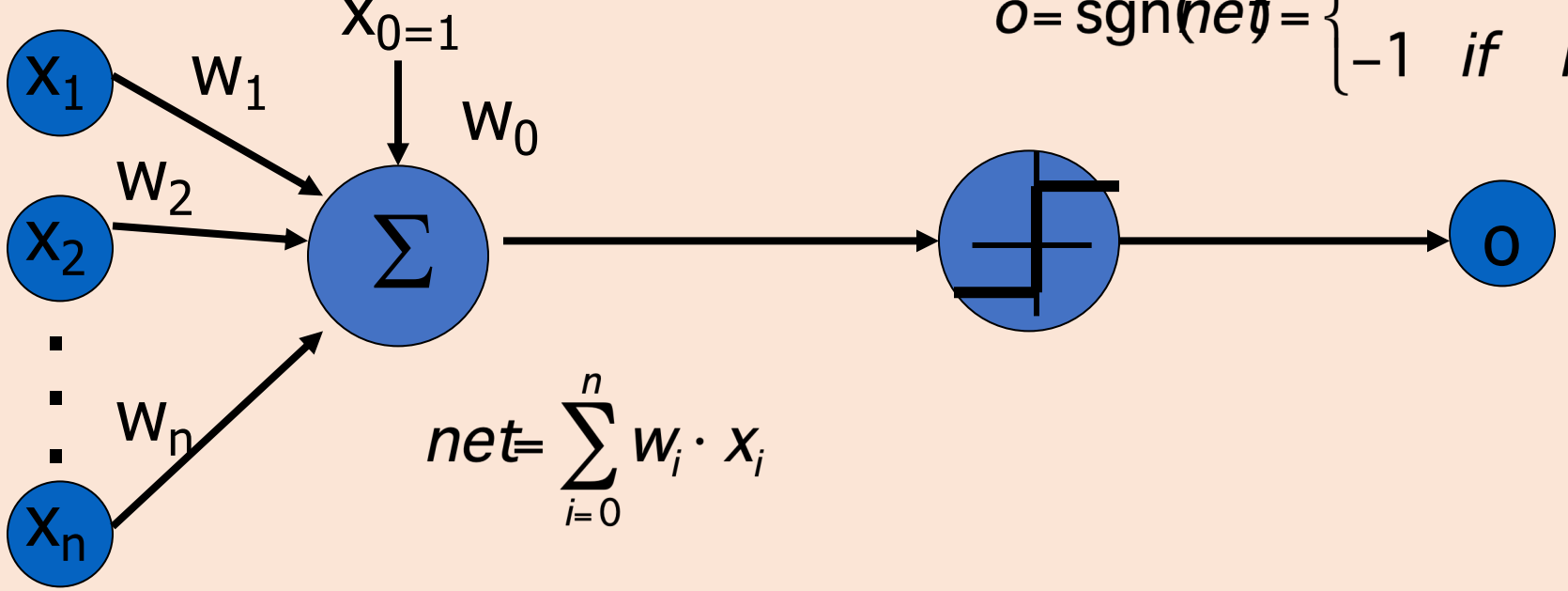
- 10^{40} Neurons
- 10^{4-5} connections per neuron



Perceptron

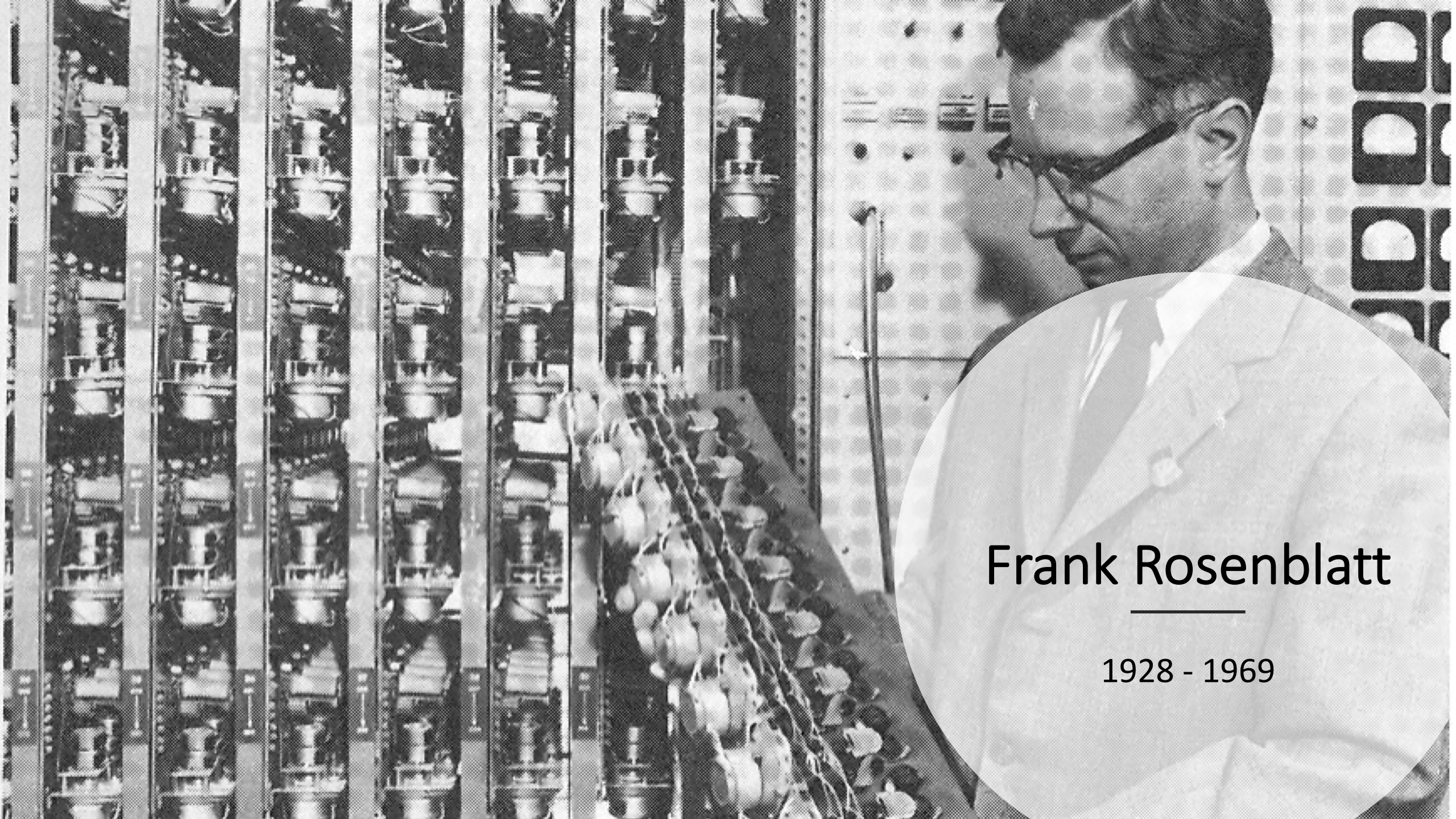
Linear treshold unit (LTU)

$$o = \operatorname{sgn}(net) = \begin{cases} 1 & if \quad net \geq 0 \\ -1 & if \quad net < 0 \end{cases}$$



$$net = \sum_{i=0}^{n} w_i \cdot x_i$$

McCulloch-Pitts model of a neuron



Frank Rosenblatt

1928 - 1969

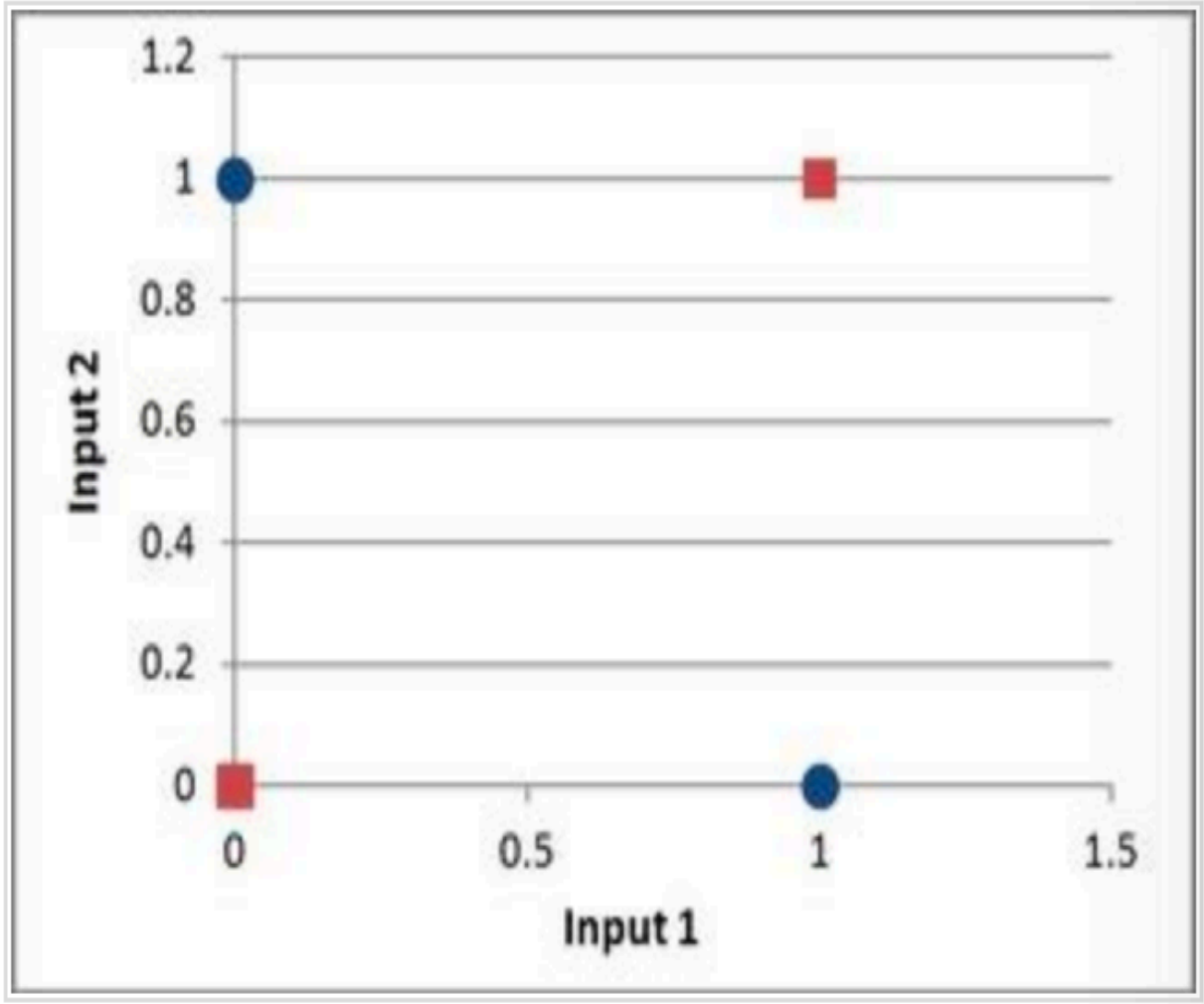
Frank Rosenblatt

- Rosenblatt's bitter rival and professional nemesis was Marvin Minsky of Carnegie Mellon University
- Minsky despised Rosenblatt, hated the concept of the perceptron, and wrote several polemics against him
- For years Minsky crusaded against Rosenblatt on a very nasty and personal level, including contacting every group who funded Rosenblatt's research to denounce him as a charlatan, hoping to ruin Rosenblatt professionally and to cut off all funding for his research in neural nets

XOR Problem

Minsky and Papert in mid 1960

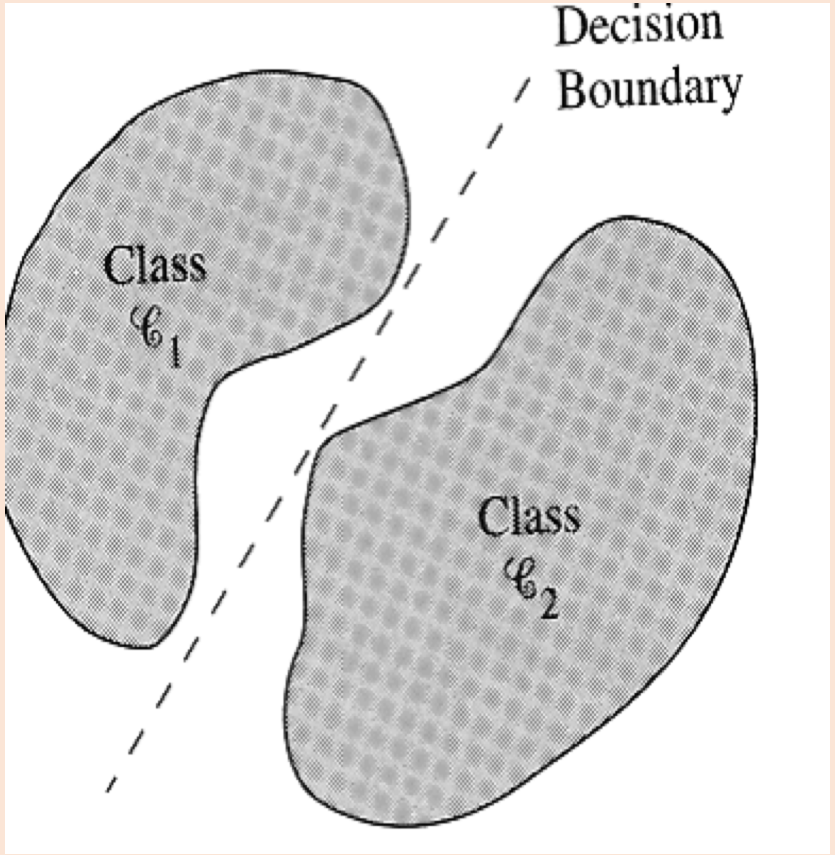
	input 1	input 2	desired output
object 1	0	0	0
object 2	0	1	1
object 3	1	0	1
object 4	1	1	0



Resemblance to real neurons

The goal of a perceptron is to correctly classify the set of pattern $\boldsymbol{D}=\{\boldsymbol{x}_1,\boldsymbol{x}_2,..\boldsymbol{x}_m\}$ into one of the classes C_1 and C_2

The output for class C_1 is *out=1* and for C_2 is *out=-1*

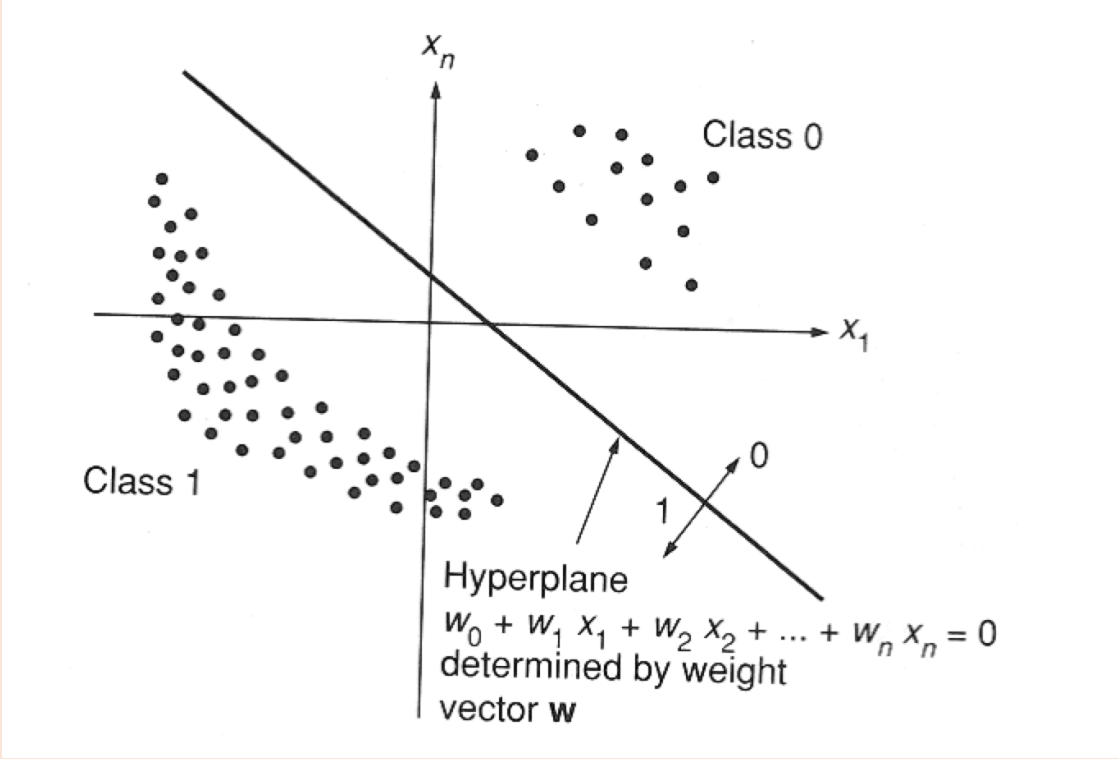


Linearly separable patterns

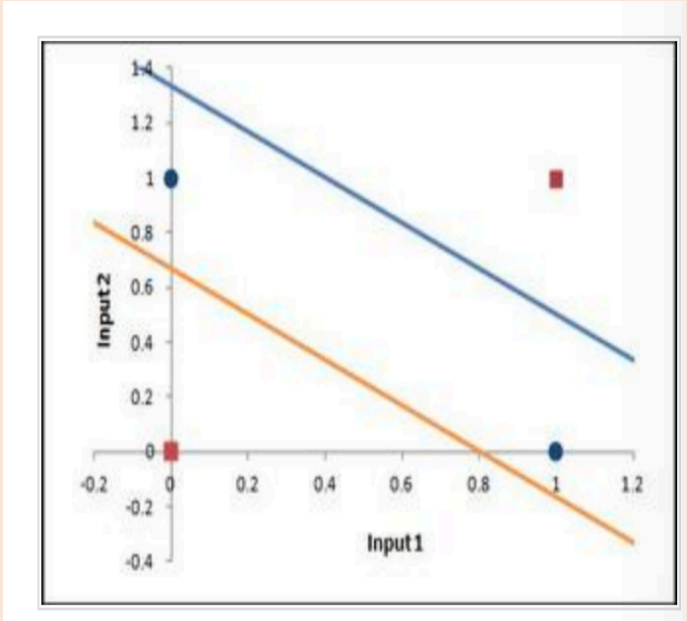
$$out = sgn\left(\sum_i w_i \cdot x_i\right)$$

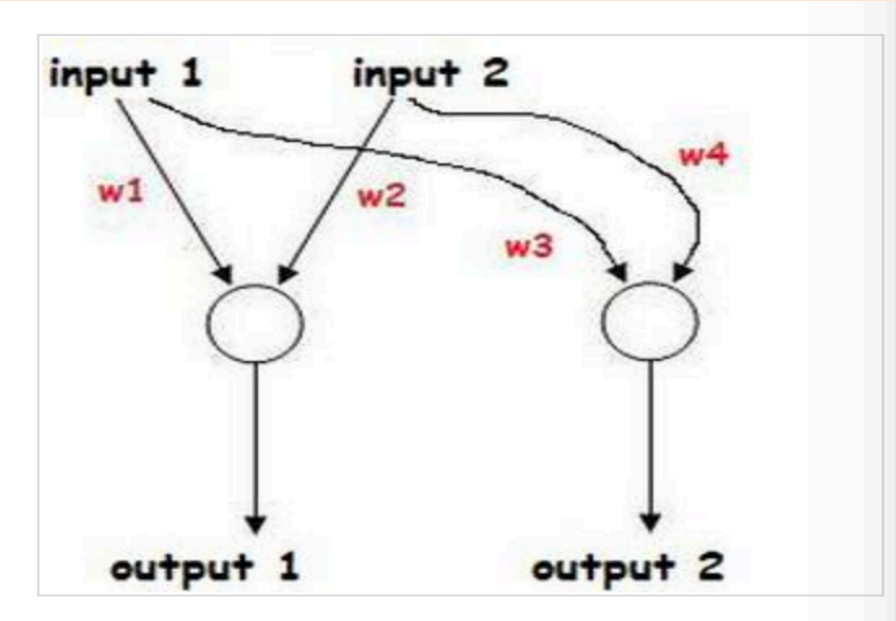
$\sum_i w_i \cdot x_i > 0$ for Class 0

$\sum_i w_i \cdot x_i \leq 0$ for class 1



Multi level networks





	input 1	input 2	output
object 1	0	0	0
object 2	0	1	1
object 3	1	0	1
object 4	1	1	1

	input 1	input 2	output
object 1	0	0	0
object 2	0	1	0
object 3	1	0	0
object 4	1	1	1

	input 1	input 2	output
object 1	0	0	0
object 2	1	0	1
object 3	1	0	1
object 4	1	1	0

we use the logic table that we've just made for the

Perceptron learning rule

- Consider linearly separable problems
- How to find appropriate weights
- Look if the output pattern *o* belongs to the desired class, has the desired value *d*

$$w^{new} = w^{old} + \Delta w \qquad \Delta w = \eta(d - o)$$

- η is called the **learning rate**
- $0 < \eta \leq 1$

Perceptron learning rule

- The algorithm converges to the correct classification
  - if the training data is linearly separable
  - and η is sufficiently small

- When assigning a value to η we must keep in mind two conflicting requirements
  - Averaging of past inputs to provide stable weights estimates, which requires small η
  - Fast adaptation with respect to real changes in the underlying distribution of the process responsible for the generation of the input vector ***x***, which requires large η

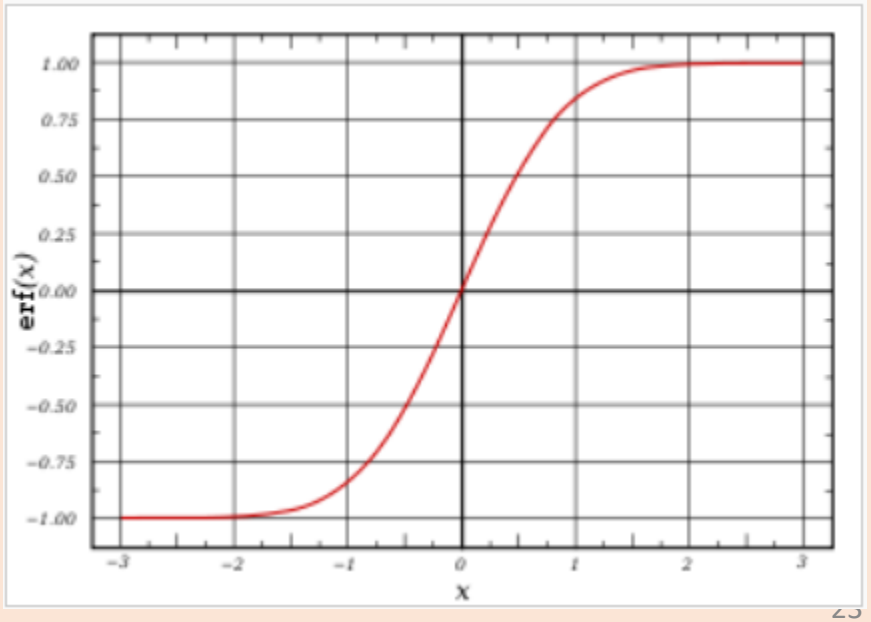
Logistic Regression

Let X be the data instance, and Y be the class label: Learn P(Y|X) directly

Let W = (W1, W2, … Wn), X=(X1, X2, …, Xn), **WX** is the dot product

Sigmoid function:

$$P(Y = 1|X) = \frac{1}{1 + e^{-<x,w>}}$$



Logistic Regression

In logistic regression, we learn the conditional distribution P(y|x)

Let $p_y(x;w)$ be our estimate of P(y|x), where w is a vector of adjustable parameters.

Assume there are two classes, y = 0 and y = 1 and

$$p_0(\mathbf{x};\mathbf{w}) = 1 - \frac{1}{1+e^{-\mathbf{wx}}} \qquad p_1(\mathbf{x};\mathbf{w}) = \frac{1}{1+e^{-\mathbf{wx}}}$$

This is equivalent to $$\log \frac{p_1(\mathbf{x};\mathbf{w})}{p_0(\mathbf{x};\mathbf{w})} = \mathbf{wx}$$

That is, the log odds of class 1 is a linear function of x

24

Learning Algorithm

The conditional data likelihood is the probability of the observed Y values in the training data, conditioned on their corresponding X values. We choose parameters w that satisfy

$$\mathbf{w} = \arg\max_{\mathbf{w}} \prod_{l} P(y^l \mid \mathbf{x}^l, \mathbf{w})$$

where **w** = $<w_0, w_1, \ldots, w_n>$ is the vector of parameters to be estimated, y^l denotes the observed value of Y in the *l* th training example, and $\mathbf{x}^l$ denotes the observed value of **X** in the l th training example